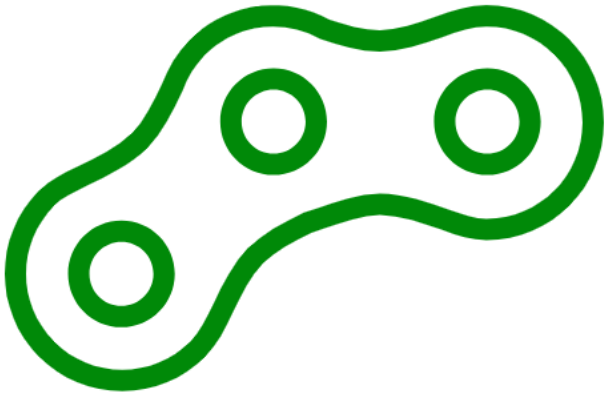


2WHEELS

Nicolò Sola

INDICE:

1. Traccia del progetto

Il prodotto 2Wheels rappresenta un sito di e-commerce che si occupa della vendita di capi d'abbigliamento per i motociclisti e prodotti per la cura delle loro motociclette. I principali motivi che mi hanno portato a scegliere di realizzare questo progetto sono stati la mia passione per le moto e la volontà di realizzare un applicativo web che trattasse questo tema. Data la ampia di libertà di scelta della traccia ho scelto di sviluppare un e-commerce dove si vendono esclusivamente prodotti per i motociclisti.

Un utente anonimo, all'interno della piattaforma, non può compiere la maggior parte delle operazioni concesse ad un utente registrato. Un cliente non loggato o privo di registrazione potrà solo navigare all'interno del sito e, non appena proverà a compiere una operazione a lui non consentita, verrà reindirizzato alla finestra di login o di registrazione. Una volta loggato o registrato, potrà accedere a tuti i servizi a lui dedicati, a seconda che egli sia un cliente oppure un fornitore.

Per usufruire di questo sito, quindi, l'utente deve per forza effettuare il login e, nel caso non disponga di account, deve provvedere alla registrazione all'interno della piattaforma. Un utente al momento della registrazione può decidere se registrarsi come cliente oppure come fornitore. Le differenze tra questi due gruppi di clienti sono molteplici:

- Cliente: Può acquistare un prodotto tra quelli disponibili, scegliendo anche tra prodotti consigliati. Una volta aggiunti al carrello tutti i prodotti di suo interesse, può procedere al pagamento che può avvenire sia utilizzando un pagamento sicuro come PayPal, sia utilizzando la carta di credito. Il cliente quindi è un utente standard, dotato di profilo, che può fare acquisti. I prodotti selezionati verranno aggiunti al carrello, il cliente potrà quindi procedere al pagamento, usufruendo anche il servizio PayPal e una volta avvenuto l'ordine, verrà processato il pagamento e verrà svuotato il suo carrello.
- Fornitore: Il fornitore è un utente che può compiere operazioni come caricare prodotti sulla piattaforma per permetterne l'acquisto ai clienti. Il fornitore ha anche la possibilità di rimuovere gli articoli caricati. Al momento del caricamento il fornitore deve inserire nome del prodotto, categoria, immagine e descrizione.

Entrambi gli utenti dispongono di un profilo dove possono vedere i prodotti ordinati, nel caso in cui l'utente sia un cliente, oppure i prodotti caricati nel caso l'utente sia un fornitore.

In seguito al login dunque, un cliente può navigare all'interno dello store alla ricerca del prodotto a cui è interessato. Per aiutare l'utente nella navigazione, sono state previste due scorciatoie. La prima è rappresentata da una barra di ricerca, all'interno della quale il cliente può cercare i prodotti che gli interessano semplicemente digitando il loro nome. La seconda scorciatoia prevede un menù a tendina all'interno del quale il cliente può accedere a pagine dedicate alle sole categorie del prodotto. Se un cliente è interessato all'acquisto di un casco, tramite questo menù potrà accedere ad una pagina in cui sono presenti solo i prodotti di categoria "caschi". Questa possibilità permette all'utente di navigare nello store in maniera più semplice e ne favorisce l'utilizzo.

La navigazione nel sito è permessa anche al fornitore, che però non può avere informazioni sui prodotti a parte il prezzo. Al fornitore è permesso solamente consultare le caratteristiche dei prodotti da lui inseriti sulla piattaforma.

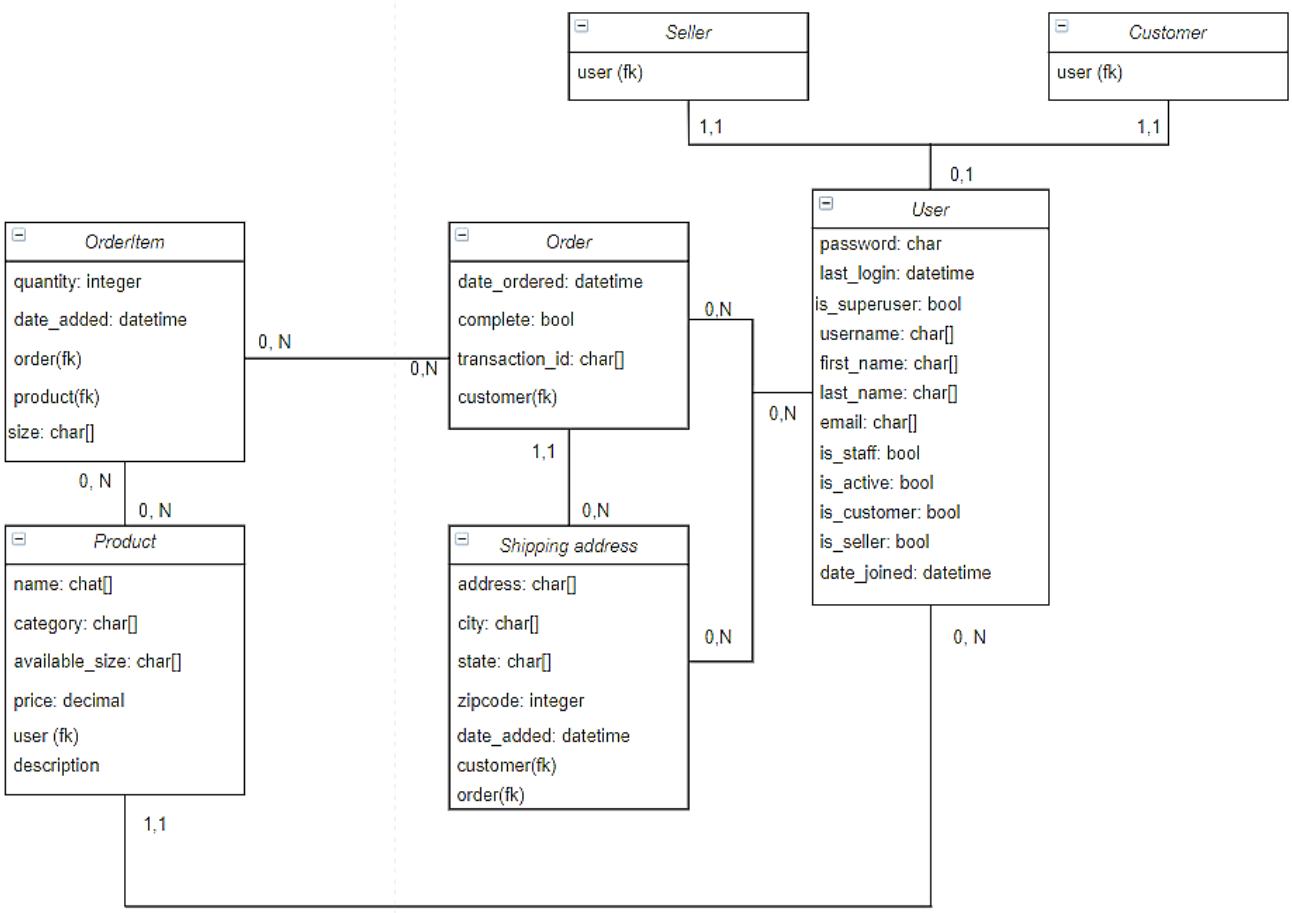
Direttamente all'interno del carrello, il cliente può rimuovere il prodotto o incrementarne la quantità, vedendo il prezzo modificarsi in tempo reale, cos' da poter analizzare da subito la quantità di denaro che dovrà investire per acquistare i prodotti che gli interessano.

Il cliente, una volta completati gli acquisti, potrà procedere al pagamento usufruendo, come detto prima, di un metodo di pagamento con transazione sicura fornito da PayPal, che gli permetterà di pagare usando PayPal, Sandbox o la sua carta di credito. Il processo di pagamento prevede la compilazione di un form dove verranno forniti i dati dell'utente come indirizzo, nome e e-mail.

Una volta completato il pagamento, il carrello verrà svuotato, arriverà una notifica di avvenuto ordine e sarà possibile per il cliente proseguire lo shopping in seguito ad un reindirizzamento alla homepage.

2. Diagrammi

2.1 Diagramma delle classi



Il database che ho deciso di utilizzare durante lo sviluppo del progetto è il database nativo di Django, ovvero SQL Lite.

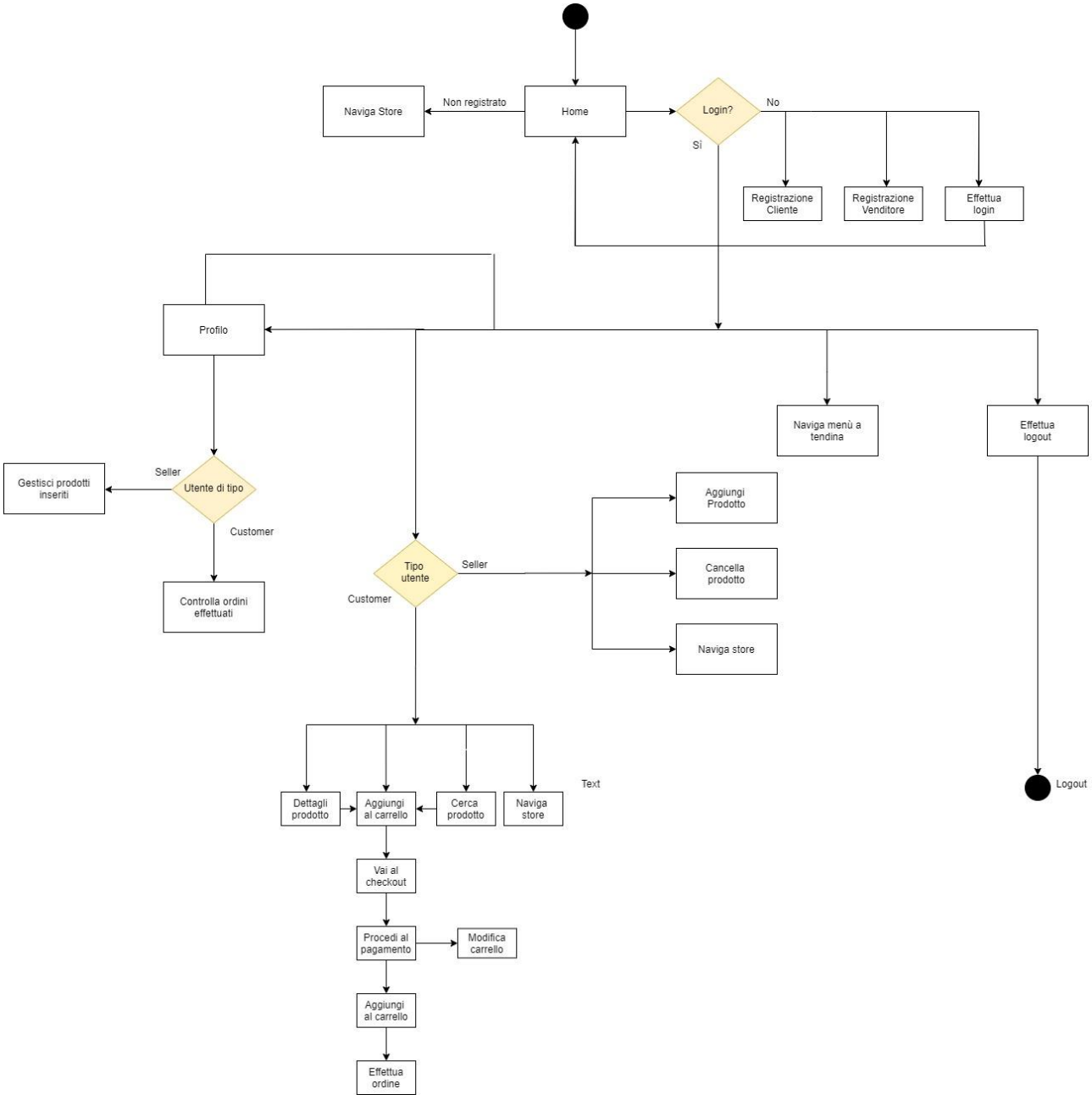
La tabella **User** si occupa di collezionare tutti i dati di tutti gli utenti che si iscrivono presso la piattaforma ed è utilizzata per l'autenticazione degli utenti stessi. Contiene inoltre i dati necessari a determinare l'entità di un utente, dato vitale per limitare l'accesso ad alcune sezioni dell'applicativo. Le tabelle **Customer** e la tabella **Seller** contengono delle Foreign Key verso la tabella User e raccolgono gli identificativi dei singoli utenti che appartengono ad ognuna di queste categorie.
La tabella **Product** e la tabella **OrderItem** contengono rispettivamente i prodotti che son contenuti all'interno dello store e i prodotti che son contenuti all'interno dell'ordine effettuato dall'utente. non a caso la tabella OrderItem contiene un riferimento sia alla tabella Product, che alla tabella Order.
Notare che la tabella **Product** è inoltre collegata alla tabella User, siccome un utente di tipo Seller, fornitore, ha la possibilità di aggiungere uno o più prodotti allo store per metterli in vendita e dunque a disposizione dei clienti.

La tabella **Order** contiene un riferimento alla tabella Customer, in modo da collegare un ordine al cliente che lo effettua. Order è inoltre collegata alla tabella OrderItem, in modo da collegare i prodotti all'interno dell'ordine, all'ordine stesso.

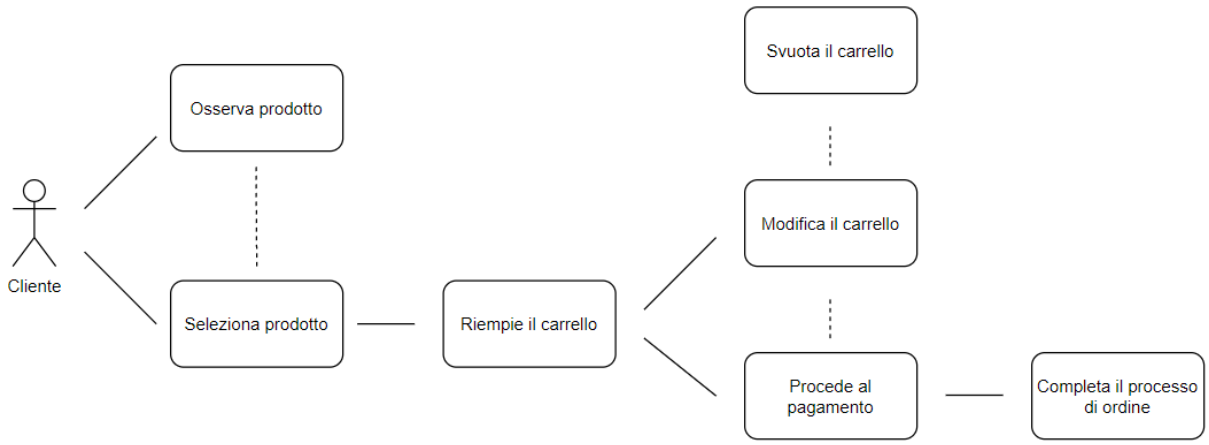
La tabella **ShippingAddress** si occupa di contenere i dati di destinazione di ogni singolo ordine in modo da poter supportare, senza stravolgere il database, una futura aggiunta della possibilità di tracciare le spedizioni e di collegare l'ordine ad un servizio di consegna a domicilio.

Ad ora, questa tabella la funzione ha il solo compito di gestire l'indirizzo dell'ordine, che viene registrato al momento del checkout da parte dell'utente, nel momento che precede l'atto del pagamento.

2.2 Diagramma di attività



2.3 Diagramma di caso d'uso che descrive il processo di acquisto di un prodotto



Le frecce tratteggiate rappresentano delle scelte opzionali (per convenzione)

3. Sezione tecnologie usate e motivazione

Le tecnologie utilizzate in prevalenza corrispondono a quelle proposte durante il corso ovvero IDE Pycharm, Python 3.8 e Django. La motivazione principale è la familiarità acquisita utilizzando questi prodotti durante il periodo delle lezioni e durante lo sviluppo del progetto svolto per il corso Progetto del software. Json è stato utilizzato assieme a Javascript per la gestione dinamica del carrello e degli ordini.

Per la realizzazione della parte di front-end sono stati utilizzati HTML e CSS, facendo largo uso del framework Bootstrap. Le tabelle sfruttano il pacchetto django-tables2 mentre i form sono renderizzati tramite "crispy".

Per il recommendation system viene usata una funzione he consiglia all'utente 3 prodotti in base alla distanza tra il nome del prodotto che si sta osservando e i nomi di tutti gli altri prodotti disponibili sullo store. Si è usufruito della similarità di levenshtein, fornita dal pacchetto "textdistance". La distanza di Levenshtein tra due stringhe A e B è il numero minimo di modifiche elementari che consentono di trasformare la A nella B. La funzione contiene un controllo sul fatto che il prodotto che viene consigliato non sia il prodotto stesso, siccome due prodotti uguali rappresenterebbero la similarità massima raggiungibile e sarebbero sicuramente consigliati.

Il codice è diviso in due applicazioni, **ecommerce** e **store** ma la maggior parte del codice, contenente tutti gli urls e i vari template sono contenuti all'interno di store.

- In store/models.py sono definiti i modelli che compongono il database su cui si basa l'applicativo.
- In store/views.py abbiamo tutte le funzioni che vengono utilizzati allo scopo di far funzionare il programma nel modo previsto.
- In store/decorators.py sono presenti dei decoratori che permettono di riservare l'accesso alle funzioni soltanto ad utenti che hanno effettuato il login e, in alcuni casi, soltanto un utente cliente, o fornitore, potrà accedere ad una determinata funzionalità, così da evitare che per esempio un fornitore possa avere un carrello o possa effettuare degli ordini.

Per il processo di login e di registrazione si è fatto uso della classe AbstractUser e viene gestito tutto d Django. L'unica modifica apportata è stata quella di convertire la lingua da inglese ad italiano. Tutto il codice usa una nomenclatura inglese, così che chiunque possa essere in grado di comprendere il codice. La lingua dell'applicativo però è italiano, essendo destinato ad utenti proveniente dall'Italia.

4. Scelte strutturali

Per evitare che un utente esterno possa modificare il prezzo a suo piacimento, viene eseguito un confronto sul prezzo in fase di back-end e se il totale non corrisponde con il prezzo proposto al momento del pagamento, il sistema si accorge dell'errore e ne impedisce il processo dell'ordine.

Per evitare che un utente non registrato possa compiere operazioni all'interno della piattaforma si è fatto uso dei decoratori per permettere solo ad utenti loggati, come cliente in certi casi o come fornitore in altri casi, di compiere operazioni a loro dedicate.
La distinzione tra utenti prevede anche il cambiamento della barra di navigazione. Solo per il cliente, per esempio, comparirà in alto il carrello contenente i prodotti da ordinare.

L'applicativo prevede dunque tre tipi di utenti, uno anonimo e due registrati alla piattaforma. Questa separazione è necessaria al fine di realizzare un sito facilmente consultabile dagli utilizzatori, con lo scopo di separare le funzioni che ogni utente può svolgere.

Il menù a tendina è stato realizzato per permettere, insieme alla funzione di ricerca del prodotto in base al nome, una migliore esperienza al cliente che vuole effettuare acquisti all'interno del sito.

Il profilo, soprattutto nel caso del fornitore, è utile siccome può consultare tutti i prodotti aggiunti da lui e, nel caso volesse, potrebbe anche eliminarli.

```
if request.user.is_authenticated:
    if request.user.is_seller:
        seller = request.user.seller
        context = {'products': products, 'form': form}
        return render(request, 'store/store.html', context)
    else:
        customer = request.user.customer
        order, created = Order.objects.get_or_create(customer=customer, complete=False)
        items = order.orderitem_set.all()
        cartItems = order.get_cart_items
else:
    # Create empty cart for now for non-logged in user
    items = []
    order = {'get_cart_total': 0, 'get_cart_items': 0}
    cartItems = order['get_cart_items']
```

In questa porzione di codice possiamo vedere come, a seconda che lo store sia visitato da un utente registrato come cliente o come venditore, lo store assuma funzionalità differenti. Il venditore non ha accesso alle funzionalità previste dal carrello, le quali sono una prerogativa del cliente.

```
@login_required()
def search_helmet(request):
    """
        Funzione che si occupa di visualizzare sullo store solo prodotti di categoria casco.

        :return:
                La pagina che prevede la visualizzazione solo di articoli di categoria casco.
    """
    products = Product.objects.filter(category='Casco').order_by('price')
    context = {'products': products}
    return render(request, 'store/helmet.html', context)
```

In questa porzione di codice possiamo vedere la funzione che si occupa della funzionalità prevista dal menù a tendina. In particolare, questa funzione si occupa del reindirizzamento alla pagina in cui vengono mostrati solo prodotti rappresentati dalla categoria ‘Casco’. Tutte le altre funzioni che si occupano di questa funzionalità riprendono la stessa struttura e si differenziano soltanto per la pagina a cui si riferiscono, data la differente categoria di prodotto alla quale devono corrispondere.

```
    all_products = Product.objects.all()
    name_products = Product.objects.get(id=id)
    nome = name_products.name
    user_products = Product.objects.filter(user__email=request.user.email)
    all_products = all_products.difference(user_products)
```

Porzione di codice della funzione three_recommended_items che si occupa dei prodotti da consigliare. Da notare “nome” che prende il nome dell’articolo che sto visualizzando.

Pagina iniziale che mostra tutti i prodotti a disposizione dell'utente sullo store, in particolare vediamo lo store a disposizione dell'utente loggato come Cliente. Lo si nota dalla disponibilità del carrello nella barra di navigazione. Nel caso di un utente non loggato o di un utente fornitore, si avrebbero solo i prodotti, senza possibilità di osservarne i dettagli o senza poter aggiungere il prodotto al carrello.

5. Unit testing

Nei test che possiamo vedere qua sotto, oppure all'interno del file tests.py, ho testato l'accesso alla pagina del profilo da parte dei vari utenti. Ovviamente un utente non registrato e loggato non ha un profilo all'interno della piattaforma. In questo modo son riuscito anche a testare indirettamente che il processo di login funzioni a dovere.

I test hanno confermato il risultato atteso, confermando il reindirizzamento nel caso in cui l'utente non sia loggato, e reindirizzando correttamente la pagina nel caso in cui l'utente abbia effettuat il login presso la piattaforma, indipendentemente dal tipo di utente.

```
class TestViews(TestCase):

    def setUp(self):
        self.customer_test = User.objects.create_user(
            username="Customer",
            password="Registration9",
            is_customer=True
        )

        self.seller_test = User.objects.create_user(
            username="Seller",
            password="Registration9",
            is_seller=True
        )

    def customerLogin(self):
        self.client.login(username="Customer", password="Registration9")

    def sellerLogin(self):
        self.client.login(username="Seller", password="Registration9")

    def test_profilo_no_logged_user(self):...

    def test_profilo_logged_customer(self):...

    def test_profilo_logged_seller(self):...
```

Ho inoltre voluto testare che il metodo get_total che si occupa di raccogliere il totale dell'ordine, moltiplicando la quantità dei prodotti per il loro prezzo, funzioni come previsto.

Dato il prezzo del prodotto e la quantità del prodotto, viene calcolato il prezzo totale e correttamente, quando il prodotto costa 200 euro e ne vengono ordinati 2 pezzi, si ha un costo totale pari a 400. Nel caso i pezzi ordinati dello stesso prodotto siano 4, il prezzo totale è di 800 euro.

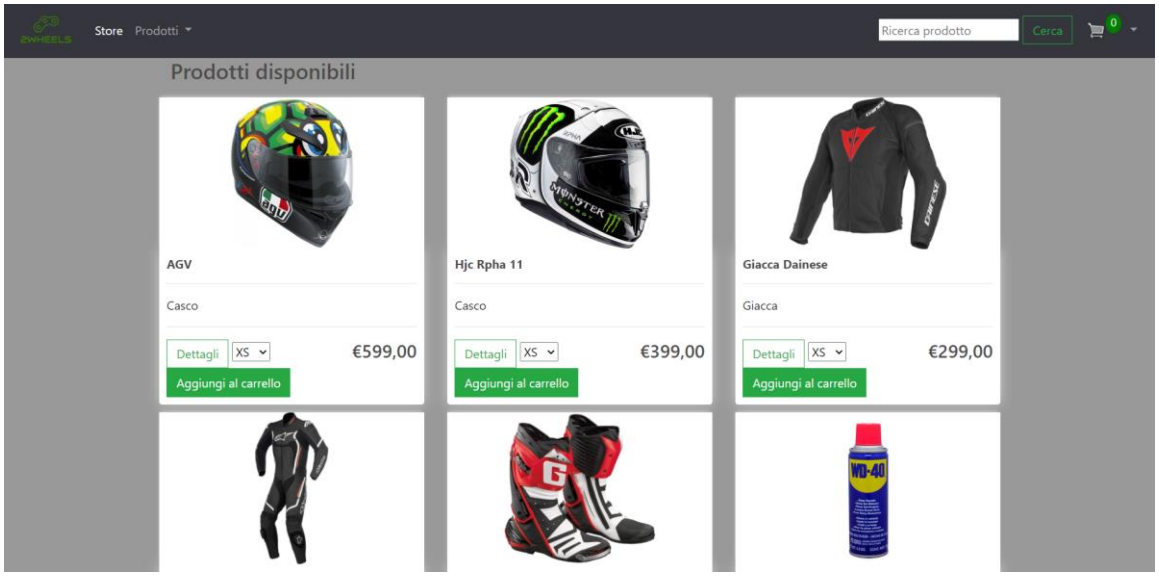
```
class TestOrderItem(TestCase):

    def test_orderitem(self):
        product = Product()
        product.price = 200
        product.save()

        order = Order()
        order.save()

        item = OrderItem()
        item.product = product
        item.order = order
        item.quantity = 2
        item.size = "XS"
        item.date_added = "2019/04/04"
        item.save()
        self.assertEquals(400, item.get_total)

        item.quantity = 4
        item.save()

        self.assertEquals(800, item.get_total)
```

Seguendo le indicazioni del professor Missiroli, può essere utile testare l'usabilità del sito facendo provare il sito ad una persona che lo utilizza per la prima volta. Il risultato è stato buono, la persona in questione ha ritenuto usabile il sito ed è riuscita a compiere tutte le operazioni senza incontrare difficoltà.

6. Screenshot del sito

In questo screenshot possiamo notare la home dedicata ad un utente loggato come cliente. Lo si nota dalla presenza del pulsante di ricerca e dal carrello nella barra di navigazione. L'icona che rappresenta il totale dei prodotti all'interno del carrello viene aggiornata dinamicamente ad ogni aggiunta del prodotto all'interno del carrello.

Nel box dedicato al singolo prodotto si può notare il pulsante aggiungi al carrello che permette al cliente di selezionare il prodotto che gli interessa e aggiungerlo al carrello dove ritroverà tutti i prodotti pronti per essere ordinati.



In questo screenshot possiamo notare la tabella che raffigura tutti i prodotti inseriti dall'utente. sfruttando la tecnologia offerta da django-tables2, la tabella risulta perfettamente ordinabile cliccando su ogni campo della tabella. Cliccando per esempio su "Nome", la tabella verrà ordinata seguendo l'ordine alfabetico dei nomi.

Nome	Categoria	Taglie disponibili	Utente	Prezzo	Descrizione	
AGV	Casco	XS, S, M, L, XL, XXL	Cabri	599,00	Casco in carbonio	Rimuovi
Hjc Rpha 11	Casco	XS, S, M, L, XL, XXL	Cabri	399,00	Casco in carbonio	Rimuovi
Giacca Dainese	Giacca	XS, S, M, L, XL, XXL	Cabri	299,00	Giacca in pelle	Rimuovi
Tuta alpinestars	Tuta	XS, S, M, L, XL, XXL	Cabri	499,00	Tuta racing in pelle	Rimuovi
Stivali Gaerne GP-1 RED	Stivali	35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47	Cabri	189,00	Stivali racing, ultra-protettivi e di ottima fattura	Rimuovi
WD-40	Manutenzione moto	—	Cabri	4,99	Lubrificante multi-uso. Una garanzia al tuo servizio	Rimuovi

7. Eventuali problemi riscontrati

Il problema principale che ho riscontrato è stato quello di realizzare il sistema di recommendation, ho avuto difficoltà a comprendere bene come implementare la funzione. Alla fine, son riuscito a realizzarne uno che si basa sulla distanza tra il nome dell'articolo di cui sto osservando i dettagli e i nomi di tutti gli altri prodotti a disposizione dell'utente sullo store. Consiglia i 3 prodotti più vicini basandosi sulla distanza di Levenshtein.

La sola cosa che mi lascia l'amaro in bocca è stata la mancanza di capacità di far leggere la taglia del capo al database, ho impostato tutto il funzionamento, ma, avendo realizzato che l'unica via percorribile era l'uso del php, ho dovuto rinunciare al completo funzionamento.

Sono comunque molto soddisfatto per l'esperienza che ho ricavato nello sviluppare un progetto simile e spero mi possa essere molto utile in futuro, in campo aziendale, essendo orientato a percorrere la scelta dello sviluppatore web.